DEEPSTREAM SDK 7.1 FOR NVIDIA DGPU/X86 (NVAIE)

RN-09353-003 | October 22, 2024
Advance Information | Subject to Change

7.1 Release Notes

TABLE OF CONTENTS

1.0 ABOUT THIS RELEASE

These release notes are for the NVIDIA® DeepStream SDK for NVIDIA® Tesla®, NVIDIA® Ampere®, NVIDIA® Hopper®, NVIDIA® Ada Lovelace®.

1.1 WHAT'S NEW

The following new features are supported in this DeepStream SDK release:

1.1.1 DS 7.1

- Supports Triton 24.08 and Rivermax v1.40/v1.50. Current release is part of NVAIE (https://www.nvidia.com/en-us/data-center/products/ai-enterprise/) and supports only x86/dGPU platforms. Release is through Triton docker: nvcr.io/nvidia/deepstream-pb24h2:7.1-triton-x86
- New Service maker framework in Python (Alpha): New application layer that removes the need to understand GStreamer application programming paradigm and enable to use Python.
- Support Gray 16 LE type.
- Postprocessing plugin to support output tensor meta from custom preprocessing
- Support Access to tensor metadata with Service Maker C++ APIs
- nvvideoconvert to support UYVY (8-bit YCbCr-4:2:2) on x86/dGPU
- Enhanced Single-View 3D Tracking.
- Improved ReID Accuracy in Tracker.
- NVIDIA TAO toolkit (previously called NVIDIA Transfer Learning Toolkit) models from https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps (branch: release/tao_ds7.1ga) integrated into SDK.
- Improved stability.
- Source code release

- Nvurisrcbin
- Message broker: protocol adapter code and nvmsgbroker API code release (amqp, azure, kafka, mqtt, redis, nvmsgbroker)

- Python bindings and samples updates:

  - Build system update: new build system using PyPA to support pip 24.2
  - Pybind11 version updated to v2.13.0
  - New bindings:

    - NvDsObjEncOutParams, NvDsObjEncUsrArgs
    - nvds_obj_enc_create_context(), nvds_obj_enc_process(), nvds_obj_enc_finish(), nvds_obj_enc_destroy_context()
    - NvDsAnalyticsObjInfo.objStatus
    - NvDsObjReid

1.2 CONTENTS OF THIS RELEASE

This release includes the following:

- The DeepStream SDK. Refer to *NVIDIA DeepStream SDK Developer Guide 7.1 Release* for a detailed description of the contents of the DeepStream release package. The Developer Guide also contains other information to help you get started with DeepStream, including information about system software and hardware requirements and external software dependencies that you must install before you use the SDK.

  - For detailed information about GStreamer plugins and metadata usage, see the "DeepStream Plugin Guide" section in the *NVIDIA DeepStream SDK Developer Guide 7.1 Release*.
  - For detailed troubleshooting information and frequently asked questions, see the "DeepStream Troubleshooting and FAQ Guide" section in the *NVIDIA DeepStream SDK Developer Guide 7.1 Release*.

- DeepStream SDK for dGPU Software License Agreement (SLA).
- `LICENSE.txt` contains the license terms of third-party libraries used.

1.3 DOCUMENTATION IN THIS RELEASE

This release contains the following documentation.

- *NVIDIA DeepStream SDK Developer Guide 7.1 Release*
- *NVIDIA DeepStream SDK API Reference*
- *NVIDIA DeepStream Python API Reference*

1.4 DIFFERENCES WITH DEEPSTREAM 6.1 AND ABOVE

`gstreamer1.0-libav, libav, OSS encoder,decoder plugins (x264/x265) and audioparsers` packages are removed in DeepStream dockers from DeepStream 6.1. You may install these packages based on your requirement `(gstreamer1.0-plugins-good/ gstreamer1.0-plugins-bad/ gstreamer1.0-plugins-ugly).` While running DeepStream applications inside dockers, you may see the following warnings:

```
WARNING from src_elem: No decoder available for type 'audio/mpeg,
mpegversion=(int)4, framed=(boolean)true, stream-format=(string)raw,
level=(string)2, base-profile=(string)lc, profile=(string)lc,
codec_data=(buffer)119056e500, rate=(int)48000, channels=(int)2'.


Debug info: gsturidecodebin.c(920): unknown_type_cb ():
```

To avoid such warnings, install `gstreamer1.0-libav and gstreamer1.0-plugins-good` inside docker.

Specifically, for deepstream-nmos, deepstream-avsync-app and python based deepstream-imagedata-multistream app you would need to install `gstreamer1.0-libav` and `gstreamer1.0-plugins-good`.

1.5 BREAKING CHANGES

- From DeepStream 7.1, the following models are deprecated.
  - Facetetect, FacedetectIR, PeopleSegnet, bodyposenet, gesturenet,emotionnet, hearratenet, gazenet, facial landmark estimation models and its corresponding applications.
  - Yolo OSS, SSD, DSSD, Yolov3, Yolov4, Yolov4-tiny, Fasterrcnn, Densenet models and it's corresponding applications.
  - deepstream-mrcnn-test: Instead use use github app. README contains instructions to use github app.
  - deepstream-segmentation-test application from SDK- Use github app. README contains instructions to use github app.
  - deepstream-segmentation Python sample application.
- DeepStream Audio support, ASR, TTS plugin deprecated. User to use Nvidia RIVA speech sdk.
- Deprecated support for uff and caffe models.
- All the models packaged in SDK are now onnx models.
- tao-converter is removed because TAO toolkit does not support tlt, uff models anymore.

- For Protocol adapters - Password through config file will be deprecated from the next release.
- Some of the open-source libraries related to Codecs have been removed so user might see some warnings as below, which can be ignored safely:

```
(gst-plugin-scanner:1433): GStreamer-WARNING **: 18:05:56.454: Failed to
load plugin '/usr/lib/aarch64-linux-gnu/gstreamer-1.0/libgstfaad.so':
libfaad.so.2: cannot open shared object file: No such file or directory
/bin/bash: line 1: lsmod: command not found
/bin/bash: line 1: modprobe: command not found
```

2.0 LIMITATIONS

This section provides details about issues discovered during development and QA but not resolved in this release.

- With V4L2 codecs only MAX 1024 (decode + encode) instances are provided. The maximum number of instances can be increased by making changes in open-source code.
- The Kafka protocol adapter sometimes does not automatically reconnect when the Kafka Broker to which it is connected goes down and comes back up. This requires the application to restart.
- If the `nvds` log file `ds.log` has been deleted, to restart logging you must delete the file `/run/rsyslogd.pid` within the container before reenabling logging by running the `setup_nvds_logger.sh` script. This is described in the "nvds_logger: Logging Framework" sub-section in the "Gst-nvmsgbroker" section in the *NVIDIA DeepStream Developer Guide 7.1 Release*.
- Running a DeepStream application over SSH (via putty) with X11 forwarding does not work.
- DeepStream currently expects model network width to be a multiple of 4 and network height to be a multiple of 2.
- Triton Inference Server implementation in DeepStream currently supports a single GPU. The models need to be configured to use a single GPU.
- For some models output in DeepStream is not exactly same as observed in TAO Toolkit. This is due to input scaling algorithm differences.
- Dynamic resolution change support is Alpha quality.
- On the fly Model update only supports the same type of Model with same Network parameters.
- Rivermax SDK is not part of DeepStream. So, the following warning is observed (`gst-plugin-scanner:33257`):

```
GStreamer-WARNING **: 11:38:46.882: Failed to load plugin '/usr/lib/x86_64-
linux-gnu/gstreamer-1.0/deepstream/libnvdsgst_udp.so': librivermax.so.0:
cannot open shared object file: No such file or directory
```

You can ignore this warning safely.

- RDMA functionality is only supported on x86 and only in x86 Triton docker for now.
- There can be performance drop from TensorRT to Triton for some models (5 to 15%). In such cases, user may want to use nvinfer plugin instead nvinferserver plugin.
- NVRM: XID errors seen sometimes when running 200+ streams on Ampere, Hopper and ADA.
- NVRM: XID errors seen on some setups with `gst-dsexample` and transfer learning sample apps.
- Sometimes during `deepstream-testsr` app execution, assertion " `GStreamer-CRITICAL **: 12:55:35.006: gst_pad_link_full: assertion 'GST_IS_PAD sinkpad)' failed`" is seen which can be safely ignored.
- For some of the models during engine file generation, error "`[TRT]: 3: [builder.cpp::~Builder::307] Error Code 3: API Usage Error`" observed from TensorRT, but has no impact on functionality and can be safely ignored.
- `deepstream-server` app is not supported with new `nvstreammux` plugin.
- TAO point-pillar model works only in FP32 mode.
- REST API support for few components (`decoder`, `preprocessor`, `nvinfer` along with stream addition deletion support) with limited configuration options. However, you can extend the functionality with the steps mentioned in SDK documentation.
- Sometimes, error "`GLib (gthread-posix.c): Unexpected error from C library during 'pthread_setspecific': Invalid argument`" is seen while running DeepStream applications.

  The issue is caused because of a bug in `glib 2.0-2.72` version which comes with `ubuntu22.04` by default. The issue is addressed in `glib2.76` and its installation is required to fix the issue (https://github.com/GNOME/glib/tree/2.76.6).

- In some cases, performance with Python sample apps may be lower than C version.
- while running `deepstream-opencv-test` app, warning "`gst_caps_features_set_parent_refcount: assertion 'refcount == NULL' failed`" observed. No impact on functionality & can be safely ignored.
- Minor perf drop observed w.r.t to DS 6.4 release with NvDCF perf configuration. In this case Setting environment variable `NVDS_DISABLE_CUDADEV_BLOCKINGSYNC=1` helps improve performance.
- For Azure, messages sent not matching with messages received on server side.
- Performance in WSL is not at par with Ubuntu system.
- Image encode not supported in WSL.
- The error "Failed to query video capabilities: Invalid argument" is observed while running DeepStream applications. You can safely ignore it.
- Numpy 2.x is not supported by PyDS.

3.0 NOTES

- Optical flow is supported only on dGPUs having Turing architecture (onwards).
- REST API commands only work after the video shows up on the host screen.
- NVIDIA® DeepStream SDK 7.1 supports TAO model-based applications (https://developer.nvidia.com/tao-toolkit). For more details, see https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps (branch: release/tao_ds7.1ga).
- On vGPU, only CUDA device memory `NVBUF_MEM_CUDA_DEVICE` supported.

Note:

- **OpenCV is deprecated by default. But you can enable OpenCV in plugins such as `nvinfer (nvdsinfer)` and `dsexample (gst-dsexample)` by setting `WITH_OPENCV=1` in the `Makefile` of these components. Refer to the component README for more instructions.**
- **When using docker make sure `libopencv-dev` package is installed inside docker if the Application requires it.**

3.1 APPLICATIONS MAY BE DEPLOYED IN A DOCKER CONTAINER

Applications built with DeepStream can be deployed using a Docker container, available on NGC (https://ngc.nvidia.com/). Sign up for an NVIDIA GPU Cloud account and look for `DeepStream` containers to get started.

After you sign into your NGC account, navigate to `Dashboard → Setup → Get API` key to get your `nvcr.io` authentication details.

As an example, you can use DeepStream 7.1 docker containers on NGC and run the `deepstream-test4-app` sample application as an Azure edge runtime module on your edge device.

The following procedure deploys `deepstream-test4-app`:

- Using a sample video stream (`sample_720p.h264`)
- Sending messages with minimal schema
- Running with display disabled
- Using message topic `mytopic` (message topic may not be empty)

Set up and install Azure IoT Edge on your system with the instructions provided in the Azure module client README file in the `deepstream-7.1` package:

```
<deepstream-
7.1_package>/sources/libs/azure_protocol_adaptor/module_client/README
```

See the Azure documentation for information about prerequisites for creating an Azure edge device on the Azure portal:

https://docs.microsoft.com/en-us/azure/iot-edge/how-to-deploy-modules-portal#prerequisites

To deploy deepstream-test4-app as an Azure IoT edge runtime module

1. On the Azure portal, click the IoT edge device you have created and click `Set Modules`.
2. Enter these settings:

   Container Registry Settings:

   ```
   Name: NGC
   Address: nvcr.io
   User name: $oauthtoken
   Password: use the password or API key from your NGC account
   ```

   Deployment modules:

   Add a new module with the name `ds`.

   Image URI:

   - For x86 dockers:

     ```
     docker pull nvcr.io/nvidia/deepstream-pb24h2:7.1-triton-x86
     ```

   Container Create options:

   - For X86:

     ```
     {
         "HostConfig": {
             "Runtime": "nvidia"
         },
     ```

```
    "WorkingDir":
"/opt/nvidia/deepstream/deepstream/sources/apps/sample_apps/deepstrea
m-test4",
    "ENTRYPOINT": [
        "/opt/nvidia/deepstream/deepstream/bin/deepstream-test4-app",
        "-i",
"/opt/nvidia/deepstream/deepstream/samples/streams/sample_720p.h264",
        "-p",

"/opt/nvidia/deepstream/deepstream/lib/libnvds_azure_edge_proto.so",
        "--no-display",
        "-s",
        "1",
        "--topic",
        "mytopic"
    ]}
```

3. Specify route options for the module:

   - Option 1: Use a default route where every message from every module is sent upstream.

```
{
      "routes": {
        "route": "FROM /messages/* INTO $upstream"
      }
    }
```

   - Option 2: Specific routes where messages sent upstream can be filtered based on topic name. For example, in the sample test programs, topic name `mytopic` is used for the module name ds:

```
{
        "routes": {
            "route": "FROM /messages/modules/ds/outputs/mytopic INTO
$upstream"
        }
    }
```

3.2 SAMPLE APPLICATIONS MALFUNCTION IF DOCKER ENVIRONMENT CANNOT SUPPORT DISPLAY

If the Docker environment cannot support display, the sample applications `deepstream-test1`, `deepstream-test2`, `deepstream-test3`, and `deepstream-test4` do not work as expected.

Workaround:

To correct this problem, you must recompile the test applications after replacing `nveglglessink` on x86 with `fakesink`. With `deepstream-test4`, you also have the option to specify `fakesink` by adding the `--no-display` command line switch.

Alternatively virtual display can be used. For more information refer to "How to visualize the output if the display is not attached to the system" section in "Quick Start Guide" section of *NVIDIA DeepStream Developer Guide 7.1 Release*.

3.3 TRITON INFERENCE SERVER IN DEEPSTREAM

Triton inference server (version 24.08) on dGPU is supported only via docker for x86.

Refer to the *NVIDIA DeepStream Development Guide 7.1 Release* for more details about Triton inference server.

Triton inference server Supports following frameworks:

Framework	Tesla	Notes / Limitations

TensorRT	Yes	Supports TensorRT plan or engine file (*.plan, *.engine)
		- Triton model config.pbtxt for TensorRT engine file format
		`platform: "tensorrt_plan"`
		`default_model_filename: "model.engine"`
		`input [...]`
		`output [...]`
		
		Triton-TensorRT backend documentation:
		https://github.com/triton-inference-server/tensorrt_backend

TensorFlow	Yes	- Supports Tensorflow 2.x (Tensorflow 1.x is deprecated)
		- Supports TF-TensorRT optimization
		- Supported model formats: *GraphDef* or *SavedModel*
		- Other TF formats such as checkpoint variables or estimators are not directly supported
		- Triton model config.pbtxt for Graphdef format
		`platform: "tensorflow_graphdef"`
		default_model_filename: "model.graphdef"
		
		- Triton model config.pbtxt for Graphdef format
		`platform: " tensorflow_savedmodel`
		"
		default_model_filename: "model.savedmodel"

		Triton Tensorflow backend documentation: https://github.com/triton-inference-server/tensorflow_backend

ONNX	Yes	- Supports ONNX model
		- Supports ONNX TensorRT optimization
		Triton model config.pbtxt for ONNX
		`platform: "onnxruntime_onnx"`
		`default_model_filename: "model.onnx"`
		`# [optional: TensorRT optimization, disabled by default]`
		`optimization { execution_accelerators { gpu_execution_accelerator : [ { name : "tensorrt" parameters { key: "precision_mode" value: "FP16" } parameters { key: "max_workspace_size_bytes" value: "1073741824" }} ] }}`
		Triton ONNXRuntime backend documentation:
		https://github.com/triton-inference-server/onnxruntime_backend

PyTorch(TorchScript)	Yes	- Support TorchScript models(file format *.pt), PyTorch model must be traced and saved as a TorchScript Model (.pt)
		Triton model config.pbtxt for TorchScript format
		`backend: "pytorch"`
		`platform: "pytorch_libtorch"`
		`default_model_filename: "model.pt"`
		`input [ { name: "INPUT0" }]`
		`output [ { name: "OUTPUT0" } { name: "OUTPUT1" }]`
		Triton Pytorch backend documentation:
		https://github.com/triton-inference-server/pytorch_backend

Python Backend	Yes	- Support Custom Triton-Python backend
		- Support Python Custom conda Execution Environment
		Triton model config.pbtxt for Python file format
		`backend: "python"`
		`default_model_filename: "model.py"`

		# [optional: custom conda env, disabled by default]
		parameters: {
		key: "EXECUTION_ENV_PATH",
		value: {string_value: "$$TRITON_MODEL_DIRECTORY/python3.6.tar.gz"}
		}
		
		Triton Python backend documentation:
		https://github.com/triton-inference-server/python_backend

Ensemble Models	Yes	- Support Triton Ensemble Models to connect multiple model inference graph
		Triton model config.pbtxt for ensemble model
		platform: "ensemble"
		input[...]
		output[...]
		ensemble_scheduling {
		step [
		{model_name: "model_A"},
		{model_name: "model_B"},
		{model_name: "model_C"},
		]
		}
		Triton Ensemble Model documentation:
		https://github.com/triton-inference-server/server/blob/main/docs/user_guide/architecture.md#ensemble-models

For more information refer to the following links:

- Triton inference server documentation entry: https://github.com/triton-inference-server/server
- Triton inference server model repository: https://github.com/triton-inference-server/server/blob/main/docs/user_guide/model_repository.md
- Triton inference server supported backends and QA: https://github.com/triton-inference-server/backend
- Triton Model TensorRT optimization for ONNX and TensorFlow: https://github.com/triton-inference-server/server/blob/main/docs/user_guide/optimization.md#framework-specific-optimization
- TensorFlow with TensorRT: https://docs.nvidia.com/deeplearning/frameworks/tf-trt-user-guide/index.html
- TensorFlow saved model: https://www.tensorflow.org/guide/saved_model#the_savedmodel_format_on_disk

Notice

Trademarks